Videojuegos

Curso de Diseño y Programación

Nº 14 5,99 euros

*Creación de una **tabla de récords***

*Usar las luces en **Blitz3D***

*Cómo **texturizar** los elementos del decorado*

14

Iberprensa
8 413042 951834 00014



Curso práctico de Videojuegos

AUTOR DE LA OBRA
Marcos Medina
DIRECCIÓN EDITORIAL
Eduardo Toribio
etoribio@iberprensa.com
COORDINACIÓN EDITORIAL
Eva-Margarita García
eva@iberprensa.com
DISEÑO Y MAQUETACIÓN
Antonio Gª Tomé
PRODUCCIÓN
Marisa Cogorro
SUSCRIPCIONES
Tel: 91 628 02 03
Fax: 91 628 09 35
suscripciones@iberprensa.com

FILMACIÓN: Fotpreim Duvial
IMPRESIÓN: Gráficas Don Bosco
DUPLICACIÓN CD.ROM: M.P.O.
DISTRIBUCIÓN
S.G.E.L.
Avda. Valdelaparra 29 (Pol. Ind.)
28108 Alcobendas (Madrid)
Tel.: 91 657 69 00

EDITA: Iberprensa
www.iberprensa.com



CONSEJERO
Carlos Peropadre

REDACCIÓN, PUBLICIDAD Y ADMINISTRACIÓN
C/ del Río Ter, 7 (Pol. Ind. "El Nogal")
28110 Algete (Madrid)
Tel.: 91 628 02 03
Fax: 91 628 09 35
(Añada 34 si llama desde fuera de España.)

DEPÓSITO LEGAL: M-35934-2002
ISBN: **Coleccionable:** 84 932417 2 5
Tomo 2: 84 932417 4 1
Obra Completa: 84 932417 5 X

NOTA IMPORTANTE:
Algunos programas incluidos en los CD de **"Programación y Diseño de Videojuegos"** son versiones completas, pero en otros casos se trata de versiones demo o trial, versiones de evaluación que Iberprensa quiere ofrecer a nuestros lectores. No se trata en ningún caso de las versiones comerciales de los programas, y las hemos incluido para dar al lector la oportunidad de conocer y probar esos programas y que así pueda decidir posteriormente si desea o no adquirir las versiones comerciales de cada uno.

Aprende divirtiéndote

Bienvenidos de nuevo a **Programación y Diseño de Videojuegos**, la primera obra coleccionable cuyo objetivo es formar al alumno en las principales técnicas relacionadas en el desarrollo completo de un videojuego.

A lo largo de la obra el lector está aprendiendo programación a nivel general y a nivel específico con ciertas herramientas y lenguajes, aprendiendo a trabajar con aplicaciones de retoque de imagen y también de diseño 3D y animación. Estamos descubriendo las aplicaciones profesionales más importantes de audio y conociendo la historia de lo que se denomina "la industria del videojuego", los últimos 20 años, los juegos que marcaron un avance, sus creadores y en general la evolución del videojuego.

Pero además, esta obra tiene un segundo objetivo, desarrollar y potenciar la creatividad del lector, nosotros a lo largo de las diferentes entregas pondremos las bases y tú pondrás tu ingenio, tu creatividad y tu capacidad de mejorar.

Nos encontramos a mitad de camino del viaje de 20 semanas que os proponemos, viaje articulado en 400 páginas y 20 CD-ROMs cuya finalidad es proporcionar las bases mínimas para después cada uno continuar su camino.

Recuerda que para alcanzar el éxito necesitas cumplir tres condiciones: que te gusten los juegos, poseer cierta dosis de creatividad y finalmente capacidad de estudio.

Una la cumples seguro.

sumario

14

PARA ENCUADERNAR LA OBRA:

- *Tapas del volumen 1 disponibles en Iberprensa. Pedidos por teléfono: 916280203*
- ***Tapas del volumen 2 ya a la venta en quioscos.***
- ***Los suscriptores recibirán las tapas en su domicilio sin cargo alguno como obsequio de Iberprensa.***

SERVICIO TÉCNICO:
Para consultas, dudas técnicas y reclamaciones Iberprensa ofrece la siguiente dirección de correo electrónico: games@iberprensa.com

PETICIÓN DE NÚMEROS ATRASADOS:
El envío de números sueltos o atrasados se realizará contra reembolso del precio de venta al público más el coste de los gastos de envío. Pueden ser solicitados en el teléfono de atención al cliente 91 628 02 03

Efectos especiales. Sistema de partículas

En esta entrega vamos a desarrollar una de las partes del videojuego que proporciona más espectacularidad a la acción y ayuda a crear un entorno más real.

Nos referimos al sistema de partículas que gestiona todos los efectos especiales de fuego, humo, chorros de luz y otros de carácter climático, como la nieve o la lluvia. Los sistemas de partículas suelen ser bastante complejos de implementar, sobre todo si se quieren buscar efectos realistas. Básicamente, los sistemas de partículas consisten en el manejo independiente de entidades, normalmente sprites, asignando a cada una de ellas unas propiedades físicas. Cada una de estas propiedades determina el comportamiento de las partículas que, unidas, forman un determinado efecto. Dependiendo del número de estas propiedades podemos obtener un sistema más o menos complejo y real. Evidentemente, a más variables mayores posibilidades.



Ejemplos del emisor de tipo 1, utilizado para el humo de los disparos.



Ejemplos del emisor de tipo 2, utilizado para simular los motores de la bionave y los ovnis.

(■) ¿CÓMO SE FABRICA UN EMISOR DE PARTÍCULAS?

Hay multitud de maneras de construir un sistema de partículas, como es normal dependiendo siempre de nuestras necesidades. Lo más estándar y útil es implementar un sistema general, a través de funciones, que admita valores para modificar las variables que gobiernan a las partículas. De esta forma, sólo tenemos que llamar a la función con unos valores determinados para conseguir diferentes efectos. En "Zone of Fighters" trabajaremos con una función que crea lo que se denomina "emisores". Los emisores son zonas cuadradas o rectangulares, de tamaño variable, que pueden estar situadas en cualquier lugar y donde colocamos las partículas para que evolucionen. El número de ellas determina la densidad del efecto. Por ejemplo, si queremos simular la nieve al caer, obtendremos mejores resultados si creamos mucha cantidad de partículas, ya que cada una de ellas representa a un copo. Aunque cada partícula puede ser también una entidad en 3D como cubos o esferas, es más normal el uso de sprites texturizados y siempre orientados hacia la cámara. Y ¿por qué sprites? Sencillamente, porque el uso de emisores de partículas conlleva la creación de muchas entidades para obtener un resultado más o menos decente y eso supone muchos polígonos y por consiguiente un mayor consumo de rendimiento. Si, por ejemplo, usamos cubos, necesitaríamos 12 polígonos por cada uno de ellos; así que para crear un emisor de 100 partículas se tendrían que dibujar 1200 polígonos. Como sabemos, un sprite es un objeto plano formado por sólo dos polígonos. Si lo orientamos hacia la cámara y le aplicamos una textura adecuada se pueden obtener resultados fantásticos y solo necesitaríamos 2 polígonos por partícula. La idea principal es



Ejemplos del emisor de tipo 3, utilizado en todas las explosiones.



Dos capturas del emisor de tipo 5, especializado en generar efectos climatológicos como la nieve.



El emisor de tipo 2 también se puede utilizar para generar el fuego de los volcanes.

crear un número inicial de partículas por medio de sprites. Luego, durante el juego, vamos modificando, actualizando y destruyendo estos sprites. Una vez que se llama a la función, ésta crea un emisor en donde evolucionan las partículas siguiendo los valores de las variables de entrada a través de ciertas operaciones matemáticas.

(■) DEFINIENDO LA ESTRUCTURA DE DATOS

Antes de crear las partículas debemos definir cada una de las variables que intervendrán en su evolución en el módulo "definiciones.bb". Como vamos a utilizar un gran número de entidades iguales y que tenemos que actualizar durante el juego, lo mejor es utilizar una estructura de datos (Ver Código 1).

Bien, en primer lugar vamos a dividir esta estructura para que se pueda comprender mejor. Por un lado, tenemos las variables que utilizaremos para el emisor y por otro las variables para cada partícula.

(■) CREANDO LAS PARTÍCULAS PARA CADA TIPO DE EMISOR

En el juego, utilizamos 6 emisores diferentes:

- **Emisor 1:** Se utiliza para el humo de los disparos de misiles y bombas de minifusión.
- **Emisor 2 y 3:** Se utilizan en las explosiones y para el motor de la bionave y los ovnis.
- **Emisor 4:** Se utiliza para el chorro de luz que emite cada bono.
- **Emisor 5:** Se utiliza para generar la nieve.
- **Emisor 6:** Se utiliza para generar la lluvia.

Debemos definir el mismo tipo de estructura y crear un número determinado de sprites para cada tipo de emisor. La creación de las partículas (sprites) para cada emisor la realizamos en la función "crear_particulas()" en el módulo "funcpantaudio.bb". Utilizamos un bucle desde 1 hasta el número total de partículas de cada emisor a partir de la variable global "Num_particulas", por ejemplo, para el emisor 1 tenemos:

```
For x = 1 To Num_particulas*3 ;
Num_Particulas=100 * 3 =
300 partículas para el emisor 1
  b.emisor = New emisor
  b\Entidad = CopyEntity(particula)
  b\Alfa# = Rnd(.2,.8)
  EntityAlpha(b\Entidad,b\Alfa#)
  EntityBlend b\Entidad,3
Next
```

Aquí, solo creamos el sprite ("particula") que asignamos a "Entidad" y, seguidamente, definimos su transparencia inicial. Los demás valores se irán creando y actualizando en la función de actualización del emisor.



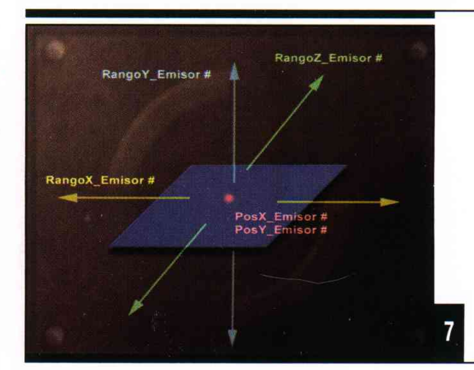
Los emisores de tipo 4 se utilizan sólo para el chorro de luz que sale de los bonos.

Podemos tener varios emisores del mismo tipo; es decir, si en el juego se producen al mismo tiempo cinco explosiones, tendremos a la misma vez cinco emisores del tipo 2 o 3. Ahora bien, si hemos creado 300 partículas para el emisor del tipo 2, en todas las explosiones que utilicen este emisor se repartirían estas partículas. Del mismo modo sucede con todos los demás tipos.

(■) GESTIONANDO LAS PARTÍCULAS DE CADA EMISOR

Ha llegado el momento de construir el núcleo de nuestro sistema de partículas. Para cada tipo de emisor utilizamos un procedimiento diferente pero con el mismo funcionamiento. Así que tomaremos como ejemplo la del emisor de tipo 1 (Ver Fig.7).

Para crear un emisor desde cualquier punto del programa llamamos a la función con los valo-



Esquema de un emisor y sus parámetros de posición y tamaño.

Código 1. Usamos una estructura de datos

```
Type emisor
  ; Emisor
  Field X_Emisor#,Y_Emisor#,Z_Emisor#    ; Posición del emisor
  Field RangoX#,RangoY#,RangoZ#          ; Tamaño del emisor
  ; Particula
  Field Entidad                          ; Entidad Sprite que forma la partícula
  Field Vida                             ; Vida de la partícula una vez emitida
  Field Alfa#                            ; Transparencia de la partícula
  Field Angulo_Rotacion#                 ; Angulo de rotación de cada partícula
  Field Xvel#,Yvel#,Zvel#                ;  Velocidad inicial de las partículas
End Type
```

res correspondientes. Por ejemplo, vamos a asignar unos valores previos y llamamos a la función pasando esos valores (Ver Código 2).

Las variables *PosX_Emisor, PosY_Emisor* y *PosZ_Emisor* corresponden a la posición del emisor de partículas. *RangoX_Emisor, RangoY_Emisor* y *RangoZ_Emisor*, definen el tamaño del emisor. *VeloX_Particula, VeloY_Particula* y *VeloZ_Particula*, asignan la velocidad inicial de las partículas. *Vida_Particula* asigna el tiempo que estará la partícula en acción. *Tamaño_Particula* y *Peso_Particula* definen el tamaño inicial que tendrá el sprite de la partícula y su peso. Y *Dir_Viento* y *Vel_Viento* asignan la dirección y velocidad que tendrá el viento.

Cada función de control de los emisores la incluimos en el módulo "funcpantaudio.bb" y su descripción la vemos en el Código 3.

Código 2. Funciones de control de los emisores

```
PosX_Emisor# = 100 :  PosY_Emisor # = 100 :PosZ_Emisor #= 100
RangoX_Emisor# = 50 : RangoY_Emisor# = 10: RangoZ_Emiso#r = 50
VeloX_Particula# = 0 : VeloY_Particula# = 0: VeloZ_Particula# = 0
Vida_Particula = 300 : Tamaño_Particula# = 5 : Peso_Particula# = 1
Dir_Viento# = 2 :  Vel_Viento# = .5
emisorparticulas (PosX_Emisor#, PosY_Emisor#,PosZ_Emisor#,
                  RangoX_Emisor#, RangoY_Emisor#, RangoZ_Emisor#,
                  VeloX_Particula#, VeloY_Particula#, VeloZ_Particula#,
                  Vida_Particula, Tamaño_Particula#, Peso_Particula#,
                  Dir_Viento#, Vel_Viento#)
```

En primer lugar, preguntamos si la partícula ha muerto a través de su variable "Vida". En caso afirmativo (*b\Vida < 0*), procedemos a inicializarla con los valores iniciales. Recordemos que no podemos destruirla, ya que sólo la creamos una vez en el programa. Por lo tanto, sólo debemos inicializarla en posición, velocidad inicial, tamaño y tiempo de vida.

Si la partícula no ha llegado al final de su vida la actualizamos según los siguientes procesos:

A)

```
b\X_Emisor#=b\X_Emisor#+b\Xvel#
b\Xvel#=b\Xvel#-.001+Dir_Viento#
```

Incrementamos la posición inicial de la partícula en la dirección del eje X según la velocidad indicada en "Xvel". Además, reducimos este incremento según la dirección del viento (*.001+Dir_Viento#*) para tener en cuenta la resistencia que éste ofrece.

Código 3. xxxxxxxxx

```
Function emisorparticulas  (PosX_Emisor#,PosY_Emisor#,PosZ_Emisor#,
                            RangoX_Emisor#,RangoY_Emisor#,RangoZ_Emisor#,
                            VeloX_Particula#, VeloY_Particula#, VeloZ_Particula#,
                            Vida_Particula, Tamaño_Particula#, Peso_Particula#,
                            Dir_Viento#, Vel_Viento#)
 For b.emisor = Each emisor
  If b\Vida<0
   b\X_Emisor#=Rnd(PosX_Emisor, PosX_Emisor +RangoX_Emisor#)
   b\Y_Emisor#=Rnd(PosY_Emisor, PosY_Emisor +RangoY_Emisor#)
   b\Z_Emisor#=Rnd(PosZ_Emisor, PosZ_Emisor +RangoZ_Emisor#)
   b\Xvel#=Rnd(-VeloX_Particula#,VeloX_Particula#)
   b\Yvel#=VeloY_Particula#
   b\Zvel#=Rnd(-VeloX_Particula#,VeloZ_Particula#)
   Escala#=Rnd(.1,Tamaño_Particula)
   ScaleSprite  (b\Entidad,Escala#,Escala#)
   b\Vida=Rnd(Vida_Particula-40,Vida_Particula+40)
  EndIf
  b\X_Emisor#=b\X_Emisor#+b\Xvel#
  b\Xvel#=b\Xvel#-.001+Dir_Viento#        ; Resistencia al viento
  b\Y_Emisor#=b\Y_Emisor#+b\YVel#
  b\Yvel#=b\Yvel#-Peso_Particula#     ; Gravedad
  b\Z_Emisor#=b\Z_Emisor#+b\Zvel#
  b\Zvel#=b\Zvel#-.001+Vel_Viento#         ; Aumenta la velocidad por el viento
  b\Vida = b\Vida - 1
  PositionEntity b\Entidad,b\X_Emisor#,b\Y_Emisor#,b\Z_Emisor#
  b\Angulo_Rotacion#=b\ Angulo_Rotacion #+2
  If b\ Angulo_Rotacion #>360 b\ Angulo_Rotacion #=1
  RotateSprite b\Entidad,b\ Angulo_Rotacion #
 Next
End Function
```



Valores necesarios para obtener un chorro de partículas estándar utilizando el programa de testeo "particulas.bb".

RangoX_Emisor#=0: RangoY_Emisor#=4: RangoZ_Emisor#=0
VeloX_Particula#=0: VeloY_Particula#=.5: VeloZ_Particula#=0
Vida_Particula=100: Tamano_Particula#=1:
Peso_Particula#=.0025
Dir_Viento#=0:Vel_Viento#=0
9

Un ejemplo de hilo de luz ideal para el humo de los misiles o la combustión de un motor.

■ B)

```
b\Y_Emisor#=b\Y_Emisor#+b\YVel#
b\Yvel#=b\Yvel#-Peso_Particula#
```

Aquí, determinamos el desplazamiento vertical de cada partícula incrementando la variable que controla el eje Y "Y_Emisor" según la velocidad dada por "Yvel". También añadimos un factor de gravedad (*.01+Peso_Particula#*) que dependerá del peso que le asignemos a la partícula.



Fuente de luz expansiva, un efecto muy llamativo que puede ser utilizado para generar explosiones de estrellas.

■ C)

```
b\Z_Emisor#=b\Z_Emisor#+b\Zvel#
b\Zvel#=b\Zvel#-.001+Vel_Viento#
```

El siguiente paso es determinar el desplazamiento en el eje Z ("Z_Emisor") según el valor de la velocidad inicial de la partícula almacenada en "Zvel". Añadimos también un factor extra como la velocidad del viento ("Vel_Viento#")

■ D)

```
b\Vida = b\Vida - 1
```

Decrementamos el valor de la vida de la partícula para dar paso a su siguiente estado.

■ E)

```
PositionEntity b\Entidad,
b\X_Emisor#,b\Y_Emisor#,b\Z_Emisor#
b\Angulo_Rotacion#=
b\ Angulo_Rotacion #+2
If b\ Angulo_Rotacion #>360
b\ Angulo_Rotacion #=1
RotateSprite b\Entidad,
b\ Angulo_Rotacion #
```

Y por último, dibujamos el sprite y lo rotamos según el ángulo definido por "Angulo_Rotacion#".

Como se puede observar no es realmente un procedimiento complicado. Resulta sencillo y además es bastante eficaz.

(■) CREANDO EFECTOS ESPECIALES

Una vez que hemos completado nuestro sistema de partículas, es el momento de disfrutarlo. Para obtener diferentes efectos basta con pasar a la función distintos valores. Para comprobar el funcionamiento de la función vamos a realizar pruebas con un pequeño programa de testeo incluido en "Extras" del CD, llamado "Particulas.bb". En él utilizamos exactamente el mismo sistema de partículas que acabamos de explicar, el que aparece en el juego. La única diferencia son las medidas de los valores, ya que creamos un entorno 3D estándar. Así que los valores que debemos colocar deben ser mil veces menor es que los utilizados en el juego.

Los diferentes efectos que podemos crear dependerán de los valores que asignemos a los parámetros de la función. Es fácil utilizar el ejemplo que se acompaña para hacer pruebas con distintos valores. Apuntando estos valores podemos tener una buena librería de efectos que podemos aplicar en nuestros juegos.

El gestor de partículas mantiene una estructura abierta. Por lo tanto, es posible la ampliación de nuevas posibilidades, como por ejemplo, tener en cuenta la elasticidad o peso de cada partícula, necesario para implementar el bote con el suelo u otras entidades.



Ejemplo de emisor en llamas. Es necesario aumentar el tamaño de las partículas y mantener una distancia cercana entre ellos para conseguir este efecto.



Ejemplo de llama de fuego inclinada por el efecto del viento.

En el próximo número...

... daremos vida al terreno de juego, colocando todo el decorado y seres vivos.

Fabricando los elementos del juego (V)

En el número anterior, terminamos de modelar los elementos que componen el decorado y ahora queda texturizarlos.

En esta ocasión, necesitaremos usar varios programas para llevar a cabo esta labor. En primer lugar, fabricaremos las plantillas necesarias que nos sirvan de guía en Paint Shop Pro. Para ello, utilizaremos el programa LithUnwrap. También, usaremos la aplicación de diseño de entornos Bryce (cualquier versión es válida) que nos ayudará a obtener texturas más realistas.

CREANDO LA PLANTILLA DE LOS ALMACENES

Antes de entrar en Paint Shop Pro para dibujar las texturas, debemos tener una plantilla o template del mapeado de los almacenes. Esta plantilla nos ayudará a tener una referencia de por dónde dibujar y así conseguir correctamente cubrir el modelo. Ejecutamos LithUnwrap y cargamos el modelo del primer almacén "almacen1.3ds" (en "Extras" del CD se pueden hallar todos los modelos de esta entrega). Antes de nada, debemos crear el mapeado UV. Seleccionamos todos los vértices con la opción *Select All / Select*. A continuación, elegimos la opción *Box* en *Tools/UV Mapping*. Y en la ventana flotante de diálogo *Box Mapping* dejamos seleccionado solo *Apply spacing*. Obtendremos una plantilla con la vista de las seis partes del almacén. Seleccionamos el tamaño que tendrá la imagen de la plantilla. En *Preferences* del menú *Options* nos situamos en la pestaña *Template* y en *Bitmap dimensions* seleccionamos un tamaño de 512 x 512. Para salvar la plantilla, elegimos la opción *Save* en *File/Template*. El programa nos volverá a preguntar las dimensiones de la imagen, que observamos que es 512 x 512. Para el siguiente almacén ("almacén2.3ds") realizamos las mismas operaciones (Ver Fig. 1).

DIBUJANDO LAS TEXTURAS DE LOS ALMACENES

Antes de pintar el aspecto de nuestro primer almacén precisamos crear cada una de las texturas que lo compondrá. Existen multitud de librerías gratuitas de imágenes para utilizar en nuestras texturas. Sin embargo, la realización manual de texturas desde cero es tarea obligada para tener un total control de nuestros diseños.

TEXTURA PARA LAS PAREDES

Vamos a empezar creando la textura de ladrillos que forman las paredes. Posteriormente, este dibujo se ajustará a la plantilla del modelo como si de un mosaico se tratara para conformar la textura final. En el aspecto que diseñamos para los almacenes, elegimos unas paredes creadas con ladrillos y con una línea de piedras en la base y bordes del techo. Empezamos, creando las paredes de ladrillo con Paint Shop. Ejecutamos el programa y seleccionamos un documento nuevo de 1024 x 1024 con color negro de fondo. Vamos a rellenar esta ima-

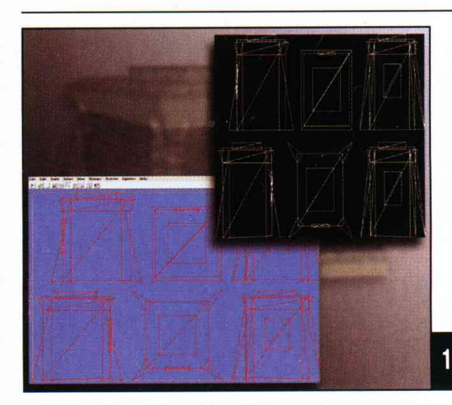
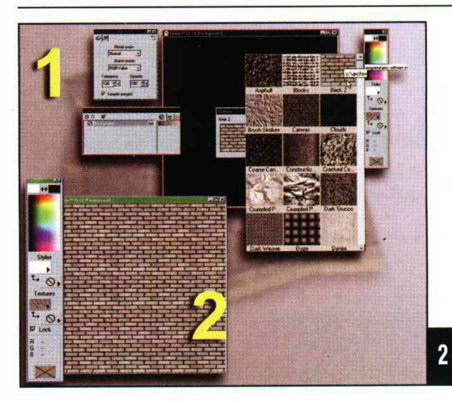
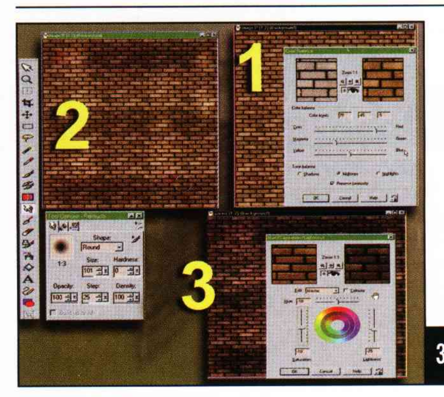


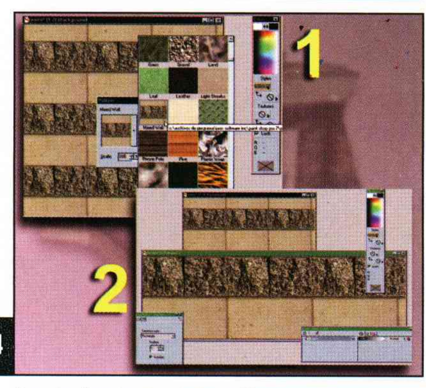
Imagen del template (plantilla) creado por LithUnwrap con las seis caras del almacén.



Aprovechemos la librería de patrones y texturas de Paint Shop Pro para obtener una pared de ladrillos.

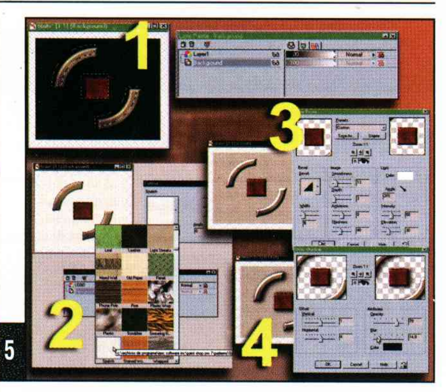


Para personalizar nuestra pared se pueden aplicar efectos de iluminación con la herramienta de retoque.



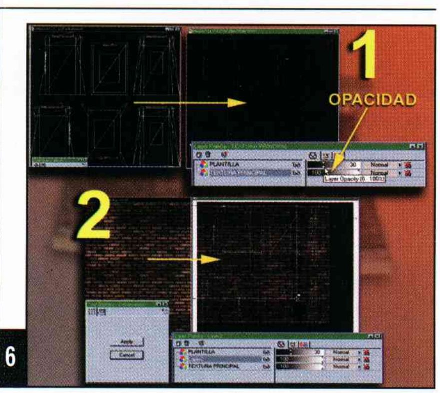
4

Para los bordes de piedra utilizaremos de nuevo la librería de patrones.



5

Debemos aislar el logotipo del fondo para poder aplicar los efectos 3D que le conferirá volumen.



6

Para realizar la textura final utilizaremos la plantilla en otra capa semitransparente como guía. Esta técnica es parecida a las utilizadas en los dibujos animados convencionales.

gen con una textura estándar de la librería de Paint Shop Pro. En la paleta de herramientas *Colors* nos situamos en *Styles* y elegimos un color sólido blanco para la tinta y anulamos el fondo (null). Luego, nos situamos en *Textures* y elegimos para la tinta la textura *Brick 2*de la librería de Paint Shop Pro.
Seguidamente, rellenamos la imagen con la herramienta de relleno *Flood Fill* (tecla F) (Ver Fig. 2).

El siguiente paso será dar un aspecto más real a los ladrillos aplicando algunas funciones de retoque fotográfico. En primer lugar, vamos a cambiar la tonalidad de la imagen mediante la función de balance de color *Color Balance* en el menú *Colors*. Elegimos unos valores para los tonos medios de rojo, verde y azul de 70, -45 y –5. Seguidamente, realizamos algunos retoques en la imagen con *Retouch* del panel de herramientas (Ver Fig. 3 / 1).

En la ventana de opciones de esta herramienta *Tool Options – Retouch*, seleccionamos un tamaño (Size) máximo (255) y una forma cuadrada para el pincel de retoque. Vamos a oscurecer algunas partes de la imagen y aclarar otras con los modos de retoque *Darken RGB* y *Ligthen RGB*. Para eliminar la uniformidad de la textura vamos también a aplicar algunos retoques diferentes como un poco de iluminación con *Lightness up* o un cambio del tono con *Hue up*. Estos retoques los haremos siempre en determinadas partes de la imagen (Ver Fig. 3 / 2). Para finalizar, cambiamos los valores de tono, saturación y brillo en *Colors/ Hue/Saturation/Lightness* a Hue = -10, Saturation = -10 y Lightness = -25. Salvamos la imagen como "pared.bmp" y pasamos a desarrollar la textura de piedra que colocaremos en la base y de separación entre las paredes y la techumbre (Ver Fig. 3 / 3).

TEXTURA PARA LOS BORDES DE PIEDRA Y LOGOTIPO

Borramos la imagen con "Ctrl" + "A" y "Suprimir". En la paleta *Colors* dejamos vacía *Textures* y en *Styles* seleccionamos el patrón (*Pattern*) *Mixed Wall* de la librería. Completamos el proceso rellenando la imagen (Ver Fig. 4 / 1).

Con la herramienta de selección elegimos la porción que se muestra en la figura 4 y la copiamos en una nueva imagen ("Ctrl"+ "C" y "Ctrl"+ "V"). Salvamos esta imagen como "borde.bmp" (Ver Fig. 4 / 2). Terminamos los elementos de la pared preparando la placa con el logotipo del juego. Para ello, cargamos la imagen del logotipo del juego que diseñamos en el número 5. Si no se dispone de ella, en el CD se puede encontrar con el nombre "titulo.bmp". Una vez cargada, aislamos el logo del fondo negro con la herramienta de selección *varita mágica* ("Magic Wand"). Hacemos clic en el fondo negro de la imagen e invertimos la selección en *Selections/Invert* (Ver Fig. 5 / 1).

Posteriormente, hacemos una copia en otra capa con "Ctrl"+ "C" y "Ctrl"+ "L". En la paleta de capas (*Layer Palette*) seleccionamos el fondo y pulsamos la tecla "suprimir" para borrar el logo del fondo. Rellenamos la capa del fondo con el estilo (*Styles*) *Spatch* de la librería con un tamaño (*Scale*) de 50 (Ver Fig. 5 / 2). Lo oscurecemos un poco con la opción *Colors/ Brightness/Contrast* y pasamos a la capa del logo para aplicar un efecto *Inner Bevel* en *Effets/3D Effects* (Ver Fig. 5 / 3) y un sombra con bastante "Blur" con el efecto *Drop Shadow* (Ver Fig. 5 / 4). Salvamos con el nombre "logo.bmp".

El techo lo dibujamos en la textura principal. Así que ya estamos preparados para realizarla utilizando la plantilla que creamos antes.

TEXTURA PRINCIPAL

El paso final es crear la textura definitiva que irá en el almacén utilizando la plantilla y las demás texturas que hemos dibujado. Creamos un documento nuevo de 512 x 512 y fondo negro. Luego, cargamos la plantilla que salvamos con LithUnwrap ("almacen1.bmp"). Debemos ahora crear una nueva capa en la imagen nueva con la plantilla. Para ello, desplazamos con el ratón la capa del fondo de la imagen de la plantilla hasta el nuevo documento. El siguiente paso es variar la transparencia de la capa de la plantilla para poder ver lo que pintamos en la capa del fondo. Este proceso es similar al utilizado para realizar, de manera clásica, los dibujos animados. Para realizar este proceso cambiamos la opacidad de la capa más o menos al 30 % en el modo normal (Ver Fig. 6 / 1).

Ahora solo tenemos que ir rellenando las siluetas con imágenes para tener la textura definitiva. Pasemos a dibujar las paredes. Cargamos la textura de la pared que fabricamos anteriormente y la pasamos a la imagen que estamos trabajando. Observamos cómo resulta de mayor tamaño que su anfitrión, así que tenemos que escalarla con la herramienta de deformación (*Deformation*) hasta que cubra la pared de la primera vista del almacén en la plantilla (incluido los polares de las esquinas) (Ver Fig. 6 / 2). Esta misma capa la copiamos con "Ctrl"+ "C" y "Ctrl"+ "L" para cubrir el resto de vistas. Para mover las capas copiadas, seleccionamos la herramienta de deformación, movemos y ajustamos (Ver Fig. 7).

Sigamos ahora, situando la textura de piedra en la base y en la parte alta. De igual forma que la anterior, cargamos la textura "borde.bmp" y la copiamos a la imagen principal. Seguidamente, escalamos y ajustamos a la base de la primera vista (Ver Fig. 8 / 1).

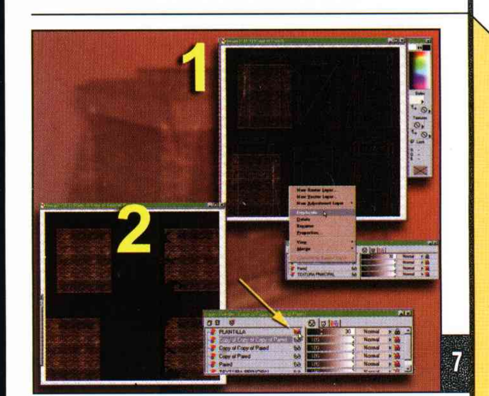
Una vez realizado este proceso, copiamos la capa y la ajustamos a otra vista y así con todas las demás. Hacemos el mismo proceso para la parte superior del edificio (Ver Fig. 8 / 2). Podemos ver el resultado ocultando la plantilla pulsando en las gafas situada al lado del nombre de la capa. Para las vistas del techo y la base utilizaremos un patrón de la librería del programa.

Nos situamos en la capa del fondo y seleccionamos con la herramienta de selección rectangular la vista del techo (la segunda). En *Styles* elegimos el patrón *Pine* y lo escalamos al 99% antes de aplicarlo. Aplicamos el patrón sobre la selección con la herramienta de relleno. Hacemos lo mismo con la vista que nos queda de la base (incluido los pilares) (Ver Fig. 9 / 1).

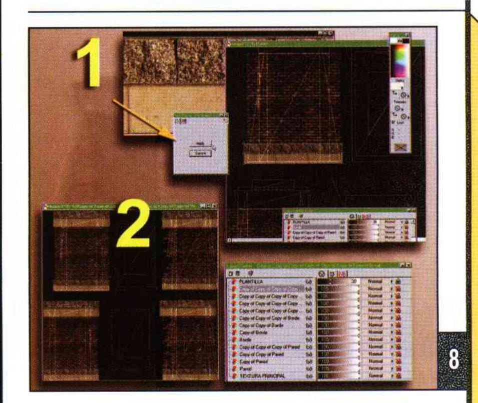
Ya solo nos queda colocar el logotipo del juego en el espacio reservado en la plantilla. Cargamos la imagen anterior del logo y la pasamos, como es habitual, a la imagen principal (Ver Fig. 9 / 2). Seguidamente, lo ajustamos al cuadrado central de las vistas tercera y sexta con la herramienta de deformación. No hay que preocuparse si la imagen queda cambiada de tamaño (Ver Fig. 9 / 3). Eso es debido a que la textura tiene forma cuadrada. Luego desde programación se ajusta automáticamente siguiendo el mapeado UV.

RETOCANDO LA TEXTURA

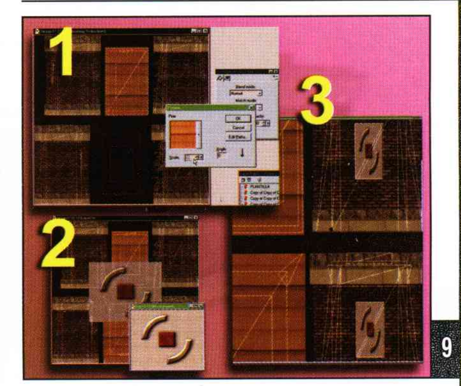
Si observamos en LithUnwrap el resultado de la textura aplicada al modelo, diremos que está correcto. Sin embargo, podemos lograr un nivel aún más detallista realizando algunos retoques extras. Si miramos con más detenimiento comprobaremos que falta algo en la textura. Realmente, da la sensación que el edificio



Después de ajustar la imagen de la pared a la primera vista de la plantilla, duplicaremos la capa y con la herramienta de transformación dibujaremos las otras vistas.



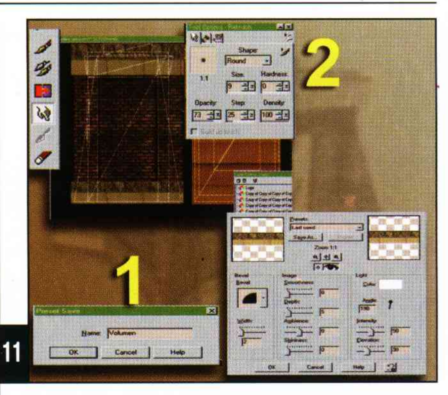
Al igual que hicimos con las paredes, es necesario escalar la imagen del borde de piedra para ajustarlo a la plantilla.



Después de rellenar las vistas del techo y la base, debemos colocar la imagen del logotipo del juego.

10

Aunque la textura parece finalizada, todavía es necesario realizar algunos retoques para crear volumen.



11

Para realzar los volúmenes siempre es conveniente aplicar ciertos tipos de efectos de luz.



12

Resultado final de nuestra textura y cómo queda aplicada al modelo.

tiene pegadas las texturas y no que forma parte del material que lo constituye. Debemos entonces aplicar ciertos efectos de iluminación para enfatizar el volumen de las formas. Así que volvamos a Paint Shop Pro para terminar el trabajo.

En primer lugar, apliquemos un efecto de relieve a las capas que conforman los bordes de piedra y el logotipo para dar la sensación de volumen. Seleccionamos la capa del borde inferior de la primera vista y elegimos el efecto 3D *Inner Bevel* en *Effects/3D Effects*. A continuación, elegimos el *Presets* custom y colocamos los siguientes valores:

- En **Bevel:** elegimos la segunda figura y un tamaño ("Width") de 2.
- En **Image:** *Smoothness* 0, *Depth* 5, *Ambience* 0 y *Shininess* 0.
- Un ángulo de 190 en **Angle**, una intensidad (*Intensity*) de 50 y una elevación (*Elevation*) de 30. Salvamos el preset en *Save as...* con el nombre "Volumen" y nos servirá para todas los demás bordes (Ver Fig. 11 / 1).

Una vez aplicado a todos los bordes el mismo efecto, pasemos a definir un poco de sombreado. Vamos a perfilar con la herramienta de retoque en el modo *Darken RGB* todos los bordes que marcan el volumen en la plantilla. Para realizar esta operación es conveniente acercar la imagen con el zoom al menos el doble para poder pintar con detalle (botón izquierdo del ratón aumenta zoom y el derecho disminuye). Pasemos a la primera vista para pintar la pared. Situémonos en la capa correspondiente y seleccionemos la herramienta de retoque. Ajustémosla a un tamaño de 5 y una opacidad (*Opacity*) del 75%. A continuación, pintamos todos los bordes con cuidado siguiendo las líneas marcadas por la plantilla (Ver Fig. 11 / 2). Haremos lo mismo en todas las vistas. El resultado final dependerá de las dotes artísticas de cada uno. Solamente hay que tener en cuenta cómo afectaría la luz al almacén en la vida real. Ahora sí podemos decir que la textura está terminada.

El segundo almacén, aunque tenga diferente forma, tiene el mismo aspecto que el primero. Por lo tanto, utiliza las mismas texturas. Lo único que tenemos que hacer es crear la plantilla y ajustar en Paint Shop Pro las texturas que tenemos de la pared y los bordes de piedra a la nueva plantilla. La única diferencia la encontramos en el techo y la base, los cuales tienen forma pentagonal. Podemos resolver esto de dos maneras. Una primera, seleccionando la vista completa con un rectángulo y rellenar con un patrón, como hasta ahora. Y una segunda, seleccionando con la herramienta de selección *Freehand* en el modo *Point to Point*.

NOTA

Se puede ir viendo el resultado de la textura directamente en el modelo a través de LithUnwrap. Solo tenemos que cargar el modelo en *File/Model/Open*, cargar la textura en *File/Material/Open* y abrir una ventana de *preview* en *Preview/Show Model* (Ver Fig. 10).



En el próximo número...

... además de terminar las texturas del decorado, aprenderemos algunas técnicas para terminar de crear las texturas para obtener texturas propias de diferentes maneras.

Nuestro primer tema musical con Anvil Studio (III)

En el número anterior aprendimos cómo preparar nuestro secuenciador para la grabación MIDI desde un instrumento exterior conectado al ordenador. En esta entrega veremos cómo editar manualmente nuestra composición desde el *Compose*.

SECCIÓN COMPOSE

Como sabemos, es necesario pasar a la sección *Compose* si queremos editar nuestras pistas. Pero antes de entrar, carguemos el tema de ejemplo "FugueGM" del CD. Una vez cargado, vamos a practicar con la primera pista. La seleccionamos y pulsamos en el botón [Compose]. Observamos cómo la pista se muestra en forma de pentagrama o staff. Podemos cambiar el modo de edición y elegir el modo *Piano Roll* pulsando en la flecha al lado del campo *Staff*. Pero de momento, seguiremos en el modo original.

Podemos elegir directamente otra pista utilizando los cursores arriba y abajo y navegar por los compases con los cursores derecho e izquierdo. Para navegar a lo largo del pentagrama con pasos más pequeños, desplazamos la mano que señala la posición actual con el ratón.

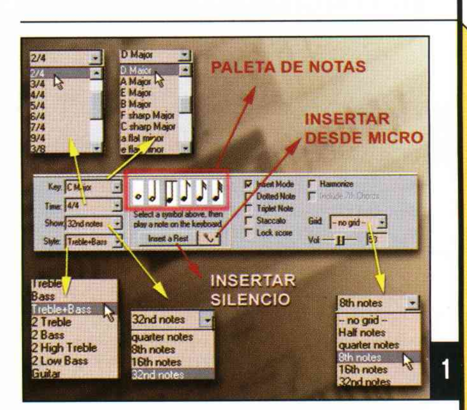
EDITANDO EL PENTAGRAMA

Vamos a hacer algunos cambios en la pista del tema que hemos cargado a través del pentagrama.

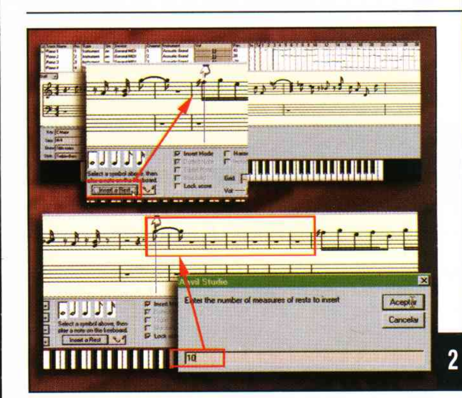
■ Opciones de pentagrama. Situado en la parte inferior del *Compose*, encontramos una serie de opciones que explicamos a continuación. A la izquierda, podemos ver cuatro casillas: *Key*, *Time*, *Show* y *Style* (Ver Fig. 1).

La primera sirve para cambiar la tonalidad de la pista La segunda, nos permite elegir el tiempo. Para ello, nos situamos en el primer compás y elegimos, por ejemplo, 2/4. Si queremos insertar un nuevo tiempo, solo tenemos que situarnos en el compás que lo llevará y elegirlo de la lista. En la casilla *Show* seleccionamos cómo queremos que se muestren las notas. Y la última casilla se utiliza para elegir el estilo que queremos que tenga el pentagrama; es decir, bajos (clave de FA), pentagrama de piano (FA y clave de SOL), doble bajo, doble agudo, etc. En la parte más a la derecha de las opciones tenemos una casilla llamada *Grid*. Aquí, Anvil Studio nos permite ver más divisiones en cada compás. Por ejemplo, si estamos en un 4/4 y seleccionamos *8th notes*, cada compás se dividirá en 8 partes (una para cada corchea).

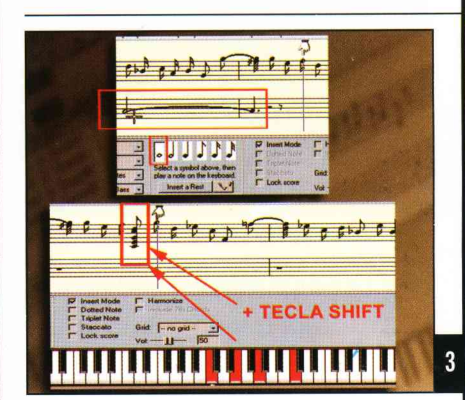
■ Insertando silencios, notas y acordes. Vamos a continuar haciendo algunos ejercicios. Comenzamos, insertando, por ejemplo, un compás en blanco a partir del primero. Para hacer esto tenemos que tener en cuenta que insertaremos un silencio equivalente a la nota que tengamos seleccionada en la paleta de notas. Así que, si estamos en un 4/4, un compás de silencio correspondería a una "redonda". Nos situamos en el segundo compás, pulsamos en la nota "redonda" y sobre el botón [Insert a Rest]. Para insertar un número determinado de compases en blanco, pulsamos dos veces sobre el botón de "insertar" y aparecerá una ventana de diálogo pidiendo el número de ellos. Para borrarlos, solo tenemos que seleccionarlos con el ratón y luego pulsar la tecla de "suprimir" del teclado (Ver Fig. 2).



Esquema de las opciones de la sección Compose.
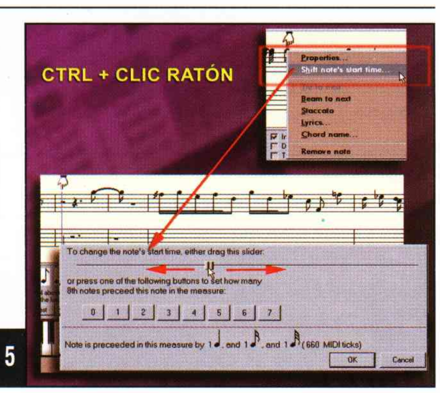


Con la opción Insert a Rest es posible insertar silencios y múltiples compases vacíos.



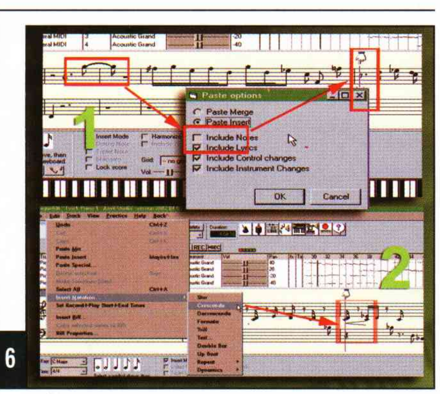
Manualmente o mediante el teclado virtual podemos insertar notas y acordes.



Pulsando con el botón derecho del ratón sobre la nota podemos elegir y modificar sus propiedades como volumen o duración.



Desplazar las notas hacia ambos lados resulta más fácil manteniendo pulsada la tecla "Ctrl".



En la zona 1 se puede observar el resultado de copiar sólo los atributos de la nota. En la zona 2 se muestra el menú de opciones de Insert Notation.

Si en vez de compases en blanco o silencios, queremos insertar una nota, solo tenemos que desplazar con el ratón cualquiera de ellas desde la paleta de notas hacia el pentagrama. El puntero cambiará a una cruz, la cual nos indica en qué posición vamos a situar la nueva nota. Una forma más fácil de insertar notas sería utilizando el teclado virtual que se muestra en pantalla. En primer lugar, seleccionamos dónde vamos a insertarla, después elegimos de la paleta el tipo de nota que queremos insertar y por último, solo tenemos que tocarla en el teclado. También es posible añadir acordes completos. Para ello, debemos dejar pulsada la tecla "Shift" mientras colocamos o tocamos las notas (Ver Fig. 3).

Otra opción interesante que Anvil Studio nos brinda es poder insertar notas desde un micrófono conectado a la tarjeta de sonido. Para realizar esta operación, activamos nuestro micro y cantamos. Seguidamente, pulsamos sobre el icono .

Cambiando propiedades. Vamos a continuar cambiando las propiedades de cualquier nota, por ejemplo, el volumen o la duración. Para abrir la ventana flotante de propiedades, pulsamos con el botón derecho del ratón sobre la nota que queremos modificar y elegimos *Properties*. En esta ventana flotante encontramos las opciones que queremos modificar (Ver Fig. 4).

Desplazando notas. Una vez insertada la nota, podemos desplazarla por el pentagrama hacia arriba o hacia abajo mediante el ratón. Para hacerlo hacia la derecha o izquierda pulsamos con el botón derecho del ratón sobre la nota para entrar en la opción *Shift note's start time*. Encontramos un deslizador (y su equivalente en botones) para asignar el desplazamiento (Ver Fig. 5).

Si queremos oír la nota mientras la desplazamos, no olvidemos dejar activada la opción mediante el icono .

Copiar grupos de notas. Para copiar notas de un lugar a otro, las seleccionamos con el ratón o mediante los cursores más la tecla "Shift" y pulsamos "Ctrl" + "C". A continuación, disponemos de tres tipos diferentes de copiado: por mezcla, por inserto o modo especial (todas estas opciones las encontramos en el menú *Edit*). El modo de mezcla (*Paste Mix*) nos permite crear un acorde; es decir, mezclamos las notas copiadas con las existentes en el lugar de destino. El modo normal (*Paste Insert*) es por inserto, en el que las notas copiadas desplazan a las notas del destino. Y por último, el modo especial (*Paste Special*) nos permite seleccionar qué evento queremos insertar en una ventana flotante de opciones. Si, por ejemplo, deseleccionamos la casilla *Include notes*, a la hora de insertar, solo lo hará el silencio que corresponde al valor de la nota (Ver Fig. 6 / 1).

Insertar anotaciones. También es posible insertar símbolos y anotaciones en el pentagrama como ligados, reguladores, *crescendos*, etc. Por ejemplo, vamos a insertar un *crescendo* en cualquier compás. Nos situamos en él y elegimos la opción *Crescendo* en *Insert Notation* del menú *Edit*. Podemos ver entonces cómo aparece el símbolo elegido (Ver Fig. 6 / 2).



TRUCO

Una manera más rápida de desplazar una nota hacia atrás o hacia delante, consiste en situarnos sobre ella y pulsar la tecla "Ctrl". Sin dejar de pulsar movemos el deslizador con el ratón directamente. Al dejar de pulsar "Ctrl" la nota se desplaza.



En el próximo número...

... aprenderemos cómo editar en el modo *Piano Roll*. Además, también empezaremos a trabajar con las pistas de audio.

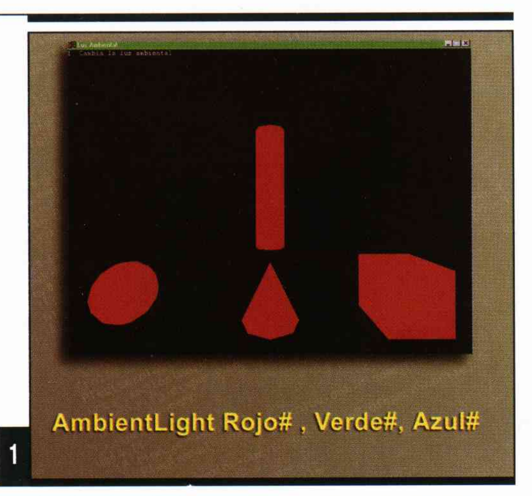
Manejo de luces en Blitz3D

Como en el mundo real, toda escena 3D necesita de una iluminación para poder ser apreciada por la cámara. Un uso adecuado de las luces puede cambiar la ambientación del entorno notablemente.

Por ejemplo, no se puede conseguir un ambiente lúgubre si no disponemos de tonos oscuros en la iluminación. Tampoco podemos simular la luz del Sol sin tonos cálidos y brillantes. Por eso, un buen criterio en la elección de las luces hará que nuestra escena 3D gane realismo.

Blitz3D nos proporciona las instrucciones necesarias para lograr la iluminación que necesitamos. Podemos crear diferentes tipos de fuentes luminosas y alterar su color, rango o potencia. Además, las luces son tratadas también como entidades, así que podemos aplicarles alguna de las funciones de éstas como posición, desplazamiento o factores de visibilidad.

En Blitz3D, las luces determinan el color de todos los vértices de un polígono. La influencia que la luz tendrá en el entorno influirá por su rango de acción.



Por defecto, Blitz3D nos proporciona una luz ambiental, la cual afecta a todos los polígonos por igual y que podemos cambiar de tono de color.

LUZ AMBIENTAL

Por defecto, Blitz3D establece una luz ambiental que ilumina a todos los vértices de los objetos por igual y que afecta a todas las demás luces. Disponemos de una instrucción que nos permite cambiar el tono de esta luz: "AmbientLight".

```
AmbientLight Componente de color Rojo#, Verde#, Azul#
```

Por ejemplo, para determinar una total oscuridad escribiremos:

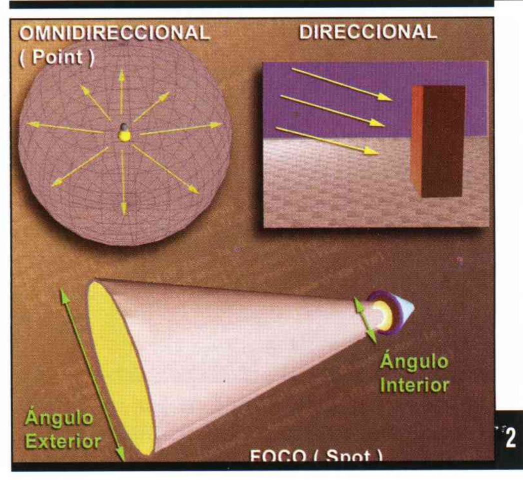
```
AmbientLight 0, 0, 0
```

Si solo usásemos esta luz, se verían todos los objetos de la escena sin ningún tipo de sombra, es decir, planos (Ver "ejemplo1.bb") (Fig. 1).

Para evitarlo, debemos crear al menos algún tipo de luz diferente.

CREANDO LUCES

En Blitz3D podemos crear tres tipos de fuentes de luz: direccional, punto de luz y foco de luz. Para crear cualquiera de estos tres tipos, disponemos de la instrucción "CreateLight":



Para evitar que los objetos de la escena se vean planos necesitamos crear algún tipo de luz. En la figura se muestra el esquema de las diferentes luces que podemos crear en Blitz3D.

```
Luz = CreateLight ( [Tipo de luz] [,entidad madre] )
```

Los tipos de luces se determinan por un número:

- **Tipo de Luz = 1** ➠ Luz direccional
- **Tipo de Luz = 2** ➠ Luz Puntual
- **Tipo de Luz = 3** ➠ Luz Focal

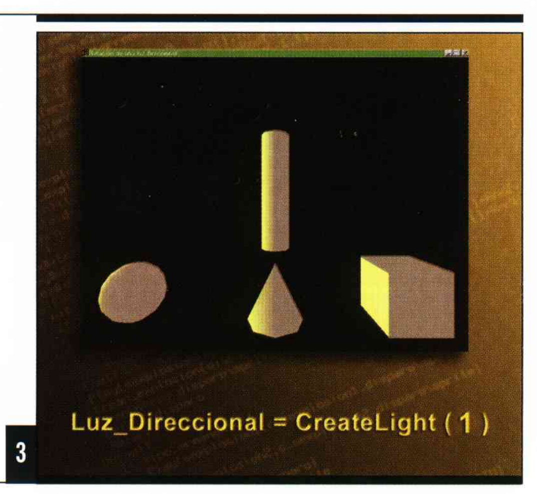
Podemos también especificar una entidad madre para la luz; es decir, si tenemos un cubo y lo asignamos como entidad madre, al mover el cubo la luz se moverá con él:

```
Cubo=CreateCube()
Luz=CreateLight(2,Cubo)
...
MoveEntity Cubo,1,0,0
...
```

Por defecto, si no asignamos ningún número al parámetro "Tipo de luz", se creará una luz direccional (Fig. 2).

LUZ DIRECCIONAL

Este tipo de fuente de luz ilumina un área completa por igual dependiendo del ángulo.



En el ejemplo número 2 podemos observar cómo afecta a los polígonos de los objetos 3D una luz direccional en rotación.

Generalmente, se utiliza para simular la luz del sol. A diferencia de la luz ambiental, la luz direccional no ilumina toda la escena sino solo los polígonos de los objetos, los cuales recibirán la luz desde la misma dirección y con la misma intensidad. Modificando el ángulo de la fuente de luz se pueden conseguir variaciones de iluminación, por ejemplo, en un plano (Ver "ejemplo2.bb") (Fig. 3):

```
Luz_direccional = CreateLight(1)
```

Este tipo de luz con un ángulo de inclinación es ideal para iluminar terrenos. El resultado de la inclinación será más visible si se activa el "shading" (sombreado) del terreno (Ver "ejemplo3.bb) (Fig. 4).

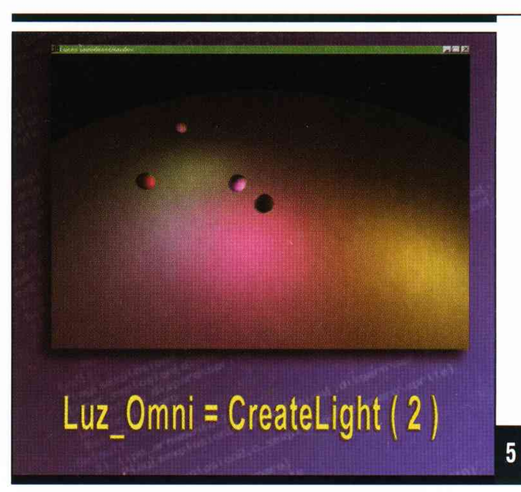
Si activamos el "shading" del terreno podemos observar cómo varía la incidencia de una luz direccional cuando cambiamos su ángulo.

LUZ OMNIDIRECCIONAL

La luz omnidireccional o puntual es un punto de luz similar, por ejemplo, a una bombilla. Se basa en crear una fuente de luz en un punto determinado, propagándose en un radio de 360 grados hasta alcanzar el límite establecido. A medida que se extiende, va perdiendo intensidad generando un degradado de luz:

```
Luz_Omni = CreateLight( 2 )
```

(Ver "Ejemplo4.bb") (Fig. 5).



Para mostrar adecuadamente el aspecto completo de las luces omnidireccionales necesitamos trabajar en una escena de gran tamaño.

FOCO DE LUZ

El tercer tipo de iluminación que podemos crear con "CreateLight" simula un proyector; es decir, un cono de luz.

```
Proyector = CreateLight ( 3 )
```

Para modificar la forma cónica se dispone de dos círculos de luz, uno interior y otro exterior, cuyo tamaño puede ser modificado. Ahora bien, este tamaño viene establecido por la apertura de ambos conos, la cual es determinada por un ángulo. Para proporcionar los valores a ambos ángulos, disponemos de la instrucción "LightConeAngles":

```
LightConeAngles Luz,
Angulo interior#,
Angulo exterior#
```

Los valores por defecto se sitúan en un ángulo mínimo de 0 y uno máximo de 90 grados. El valor máximo posible para el anillo interior va desde 0 hasta el ángulo del anillo exterior.

(Ver "ejemplo5.bb", "ejemplo6.bb" y "ejemplo7.bb") (Figs. 6, 7 y 8).

En ocasiones, necesitaremos que un foco apunte siempre hacia un objeto de la escena. Por ejemplo para iluminarlo mientras éste se mueve. Para lograrlo, tendremos que utilizar la instrucción "PointEntity":

```
PointEntity Luz, Objeto3D
```

OPERACIONES CON LAS LUCES

Los tres tipos de luces pueden posicionarse o desplazarse por la escena 3D por medio de las instrucciones "PointEntity" y "MoveEntity". Además, se les puede asignar cualquier color con la instrucción "LightColor":

```
LightColor Luz, Componente de
color Rojo#, Verde#, Azul#
```



NOTA

Es preciso recordar que Blitz3D permite crear simultáneamente todas las luces que la tarjeta de vídeo soporte; generalmente, un máximo de 7 (con la luz ambiental por defecto incluida).

6

Se pueden realizar algunos efectos artísticos usando focos.

Asignando un color blanco ("LightColor Luz, 255, 255, 255") establecemos el brillo máximo para esa luz. Por el contrario, con un color negro ("LightColor Luz, 0, 0, 0") no habrá efecto de brillo alguno. También, se puede obtener luz negativa aplicando a cada componente de color un valor negativo máximo ("LightColor Luz, -255, -255, -255"). Esto anula cualquier tipo de brillo. Es ideal para obtener efectos de sombras. En el ejemplo 8 se puede observar cómo proyectando una luz negativa desde la esfera, dicha luz origina la sombra de ésta sobre el suelo (Fig. 9).

Desgraciadamente, este tipo de técnicas no resulta eficaz en la práctica si queremos proyectar sombras irregulares o de varios objetos al mismo tiempo, ya que necesitaríamos una luz para cada uno de ellos y, lógicamente, existe un límite impuesto por nuestra tarjeta gráfica.

Como comentábamos en la introducción, la influencia de las luces viene determinada por su rango de acción. Todo lo que queda fuera de ese rango no será afectado por la iluminación. Para modificar este aspecto, Blitz3D nos proporciona la instrucción "LightRange":

```
LightRange Luz
Rango#
```

No hay establecido un valor exacto para el rango de cada luz. Éste es aproximado y pasa por aplicar el ensayo – error para obtener los resultados más adecuados.

Una opción muy interesante que nos ofrece Blitz3D es la de crear una falsa iluminación sobre objetos 3D (*meshes*). Este sistema resulta interesante para "pintar" nuestros decorados. Para realizar esta operación disponemos de la instrucción "LightMesh":



```
LightMesh Mesh, Componente de
color Rojo#, Verde#, Azul#
[,Rango][,PosX_Luz][,PosY][,PoZ]
```

UTILIZACIÓN DE LUCES

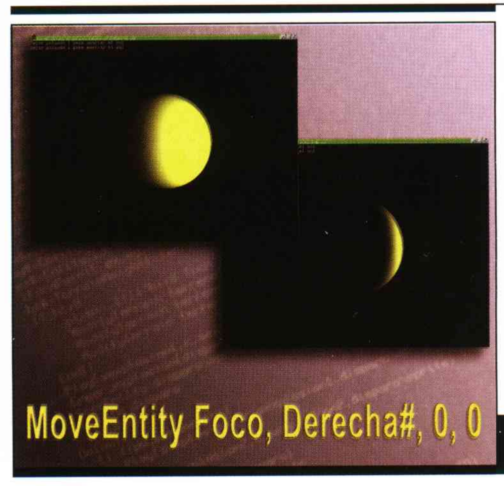
El uso de luces se torna variadísimo en multitud de ocasiones. Para los videojuegos, su utilidad es evidente y entre una lista interminable de posibilidades se me ocurren algunas interesantes y fáciles de implementar. Para ello, vamos a recorrer cada una de las diferentes instrucciones que disponemos para el desarrollo de iluminación.

NOCHE Y DÍA

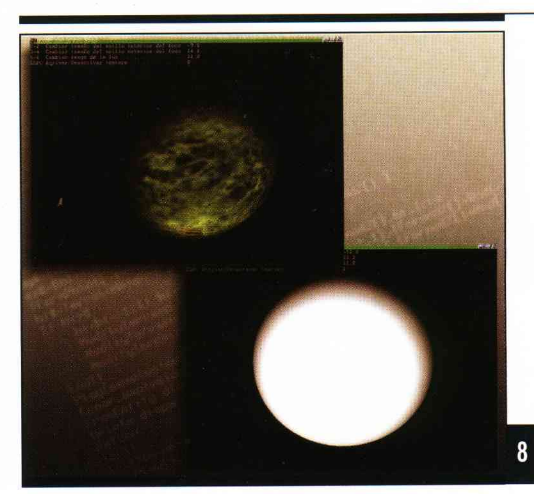
Este efecto resulta ideal para simular realismo en entornos abiertos. Es sumamente fácil de programar y el resultado "salta a la vista". El truco radica en utilizar las propiedades que nos ofrecen la instrucción "AmbientLight" y las luces direccionales. En primer lugar, es necesario crear una luz por defecto (tipo 1, direccional) que hará el papel del sol. Seguidamente, debemos utilizar una variable del tipo *float* que utilizaremos para aplicarle rotación:

```
;Definiciones
Sol = CreateLight()
Rotación_Luz# =0.0
...
```

Debemos jugar, además, con la luz ambiental, variando su brillo para aumentar el efecto de la rotación del sol. Así que tenemos que utilizar otra variable *float* para disminuir y aumentar por igual las componentes de color:



En el ejemplo 6 se aplica el desplazamiento de un foco para simular la representación de un eclipse de sol.



Podemos observar el funcionamiento de todos los parámetros de una luz focal en el ejemplo número 7 contenido en el CD.

Foco = CreateLight (3, Esfera)
LightColor Foco, -255, -255, -255

9

Por medio de luces negativas es posible obtener sombras dinámicas en tiempo real.

```
;Definiciones
Luz# = 200.0   ; Día
AmbientLight Luz#, Luz#, Luz#
```

Para lograr que la luz se modifique desde un valor a otro y viceversa, utilizaremos otra variable que servirá de incremento de la cantidad de luz (variable "Luz"). Por último, crearemos una función que hará todo el proceso y que tenemos que debemos llamar desde el bucle principal del juego:

```
Function Dia_Noche()
 If Luz<10 Variación_Luz=
 (Variacion_Luz)*-1
 If Luz>254 Variación_Luz=
 (Variacion_Luz)*-1
 Luz=Luz- Variación_Luz
 AmbientLight Luz, Luz, Luz
 Rotacion_Luz = Rotación_Luz + .1
 RotateEntity Sol, Rotación_Luz, 0,0
End Function
```



TRUCO

Se pueden utilizar las ventajas de la función coseno para lograr un modo más elegante de fluctuación entre claridad y oscuridad, evitando el uso de condicionales:

Iluminación_Máxima=50 :
Velocidad_Transición#=0.01
Luz=Iluminación_Máxima + Cos (MilliSecs())* Velocidad_Transición

Con las dos sentencias condicionales, simplemente, cambiamos el signo de "Variación_Luz" para conseguir sumar o restar el incremento del valor de "Luz". Con ello, conseguimos fluctuar de un valor a otro.

EXPLOSIONES Y DISPAROS

El segundo tipo de iluminación (omnidireccional) nos puede ayudar a enfatizar, aún más, las explosiones de nuestros juegos. Simplemente, consiste en ir creando y destruyendo luces en el momento del impacto. Realmente, si vamos a tener un uso masivo de explosiones en el juego puede ocurrir que, en ocasiones, se rebase la barrera del máximo de luces simultáneas permitido por la tarjeta de vídeo. Llegado el caso, veremos cómo nuestra escena se oscurece por momentos, ya que perdemos la luz ambiental. Para evitar este problema, es fundamental destruir la luz una vez visualizada. Aunque este sistema tiene una efectividad del 70%, siempre se puede evitar utilizar luces reales para este cometido y simularlas por medio del coloreado o pintado de los triángulos involucrados en la explosión. Esta técnica de mapeado de luces en tiempo real denominada *"vertex lighting"* resulta algo más complicada de implementar, pero es una de las alternativas más usadas por los juegos de hoy día.

De igual manera, es posible aplicar este efecto a los disparos. Básicamente, creando una luz en el momento de disparar, por ejemplo, para simular la combustión del arma. Por razones que ya hemos comentado, no es conveniente crear una luz y que se desplace en la trayectoria del disparo. Fácilmente, sobrepasaríamos el límite de luces.

LINTERNAS

Este tipo de efecto es fácil realizarlo utilizando el tercer tipo de luz (foco). Consiste en situar en el objeto emisor (linterna) un foco de luz con el ajuste del rango y el ángulo de apertura adecuados. Podíamos ver el uso de esta técnica en el tutorial que dedicamos a los niveles BSP del número 11. En ella, situamos la linterna al nivel de la cámara:

```
Camara = CreateCamera()
Linterna=CreateLight(3, camara)
LightConeAngles Linterna, 0, 60
...
```

SOMBRAS DINÁMICAS UTILIZANDO LUCES

Como hemos comentado, existe una técnica para generar sombras en tiempo real de un objeto 3D mediante el uso de luces negativas. Básicamente, la teoría consiste en asignar al objeto un foco de luz negativa (sombra), el cual se orientará hacia la fuente de luz principal (Ver "ejemplo8.bb") (Fig. 9).



En el próximo número...

... aprenderemos cómo utilizar el teclado, ratón y dispositivos para juegos en nuestros programas.



10

Ejemplo de iluminación por medio de un foco simulando una linterna.

Creación y organización de tablas de récords

Una tabla de récords es algo casi fundamental en un juego arcade, en el que el jugador acumula puntos.

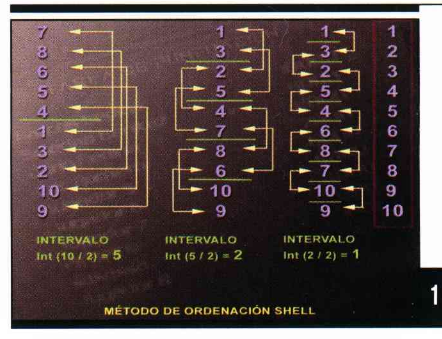
Básicamente, consiste en mostrar una lista, más o menos extensa, de nombres con los correspondientes puntos conseguidos. Para que un jugador pueda inscribir su nombre en la lista, es necesario que acumule una cierta cantidad de puntos superando una cifra mínima preestablecida. La lista puede ser todo lo grande que se quiera. Normalmente, se visualizan grupos de diez nombres en pantallas consecutivas, pero a menudo se suele usar un scroll vertical para mostrar toda la lista. En nuestro tutorial vamos a implementar una de estas tablas. La lista contendrá sólo diez jugadores y la puntuación mínima inicial para poder entrar en la tabla es de 3000. La idea es mostrar a la vez en pantalla únicamente las diez mejores puntuaciones ordenadas de mayor a menor. Cuando un jugador supere el mínimo se insertará en la tabla en el lugar correspondiente. Debemos entonces afrontar un par de problemas. Por un lado tenemos que ordenar la tabla de mayor a menor y por otro insertar un elemento nuevo si llega el caso. Para resolver el primero de los problemas se puede utilizar cualquier método de ordenación estándar que veremos a continuación.



MÉTODOS DE ORDENACIÓN

Basaremos la ordenación de la tabla en los valores de puntuación. Existen varios métodos para ordenar una tabla. Entre los más conocidos se encuentran: burbuja, selección, shell, inserción y *quicksort*. Nosotros sólo veremos dos: el método *shell* y el de burbuja.

El resultado es exactamente igual utilizando uno que otro, sin embargo, existe una relativa diferencia de velocidad entre ambos.



El método shell resulta eficaz para ordenar una tabla con muchos elementos, pero es más complicado de utilizar.

MÉTODO SHELL

Este sistema de ordenación se utiliza para tablas con un gran número de elementos. Por supuesto, no es el ideal para nuestro caso. Además, es mucho más complicado de entender que el de burbuja. Sin embargo, lo hemos incluido por motivos didácticos. Este método, básica-

Código 1. Método shell

```
Function ordenar_tabla()
 Num_Elementos=10 + 1
 While Num_Elementos>1
  Num_Elementos=Int(Num_Elementos/2)  ; Dividimos la tabla en dos partes
   SwMitad_Ordenada=0
   While SWMitad_Ordenada=0  ; Procedemos a recorrer y ordenar cada mitad
   SWMitad_Ordenada=1 ; Activamos en Switch
   For n=0 To 9-Num_Elementos ; Recorremos la primera mitad para ordenarla
    I=n+Num_Elementos ; Situamos el puntero de comparación
    If Puntuacion(n)<Puntuacion(I)
     ; Intercambiamos los elementos de la tabla para ordenarlos
     Aux_Puntuacion=Puntuacion(I): Aux_Jugador$=Jugador$(I)
     Puntuacion(I)=Puntuacion(n)  : Jugador$(I)=Jugador$(n)
     Puntuacion(n)=Aux_Puntuacion: Jugador$(n)=Aux_Jugador$
     SWMitad_Ordenada=0 ; Hasta que esta mitad no este ordenada no pasamos a la siguiente
    EndIf
   Next
  Wend
 Wend
End Function
```

2

En el ejemplo "Shell.bb" se puede observar el funcionamiento del método shell con una tabla de 20 números aleatorios.

mente, lo que hace es dividir la tabla en dos. Una vez que ha ordenado una parte, vuelve a dividir la que queda sin ordenar. Y así sucesivamente hasta que no pueda dividir más. Para crear las partes nos basamos en una variable que controla el número de elementos de cada una de ellas llamada *intervalo* (Ver "shell.bb") (Código 1).

La variable "Num_Elementos" es la que utilizamos como intervalo y "SWMitad_Ordenada" es un Switch que nos avisa si la parte está totalmente ordenada.

MÉTODO DE BURBUJA

Este método es mucho más lento para ordenar gran cantidad de elementos, pero resulta ideal para nuestra tabla de récords, ya que sólo tenemos que ordenar diez elementos. Además, es muy sencilla de comprender, así que será el que utilicemos para nuestro juego. Consiste en comparar cada elemento con todos los demás. Si están desordenados, se intercambian en cada comparación para ir situando el valor mayor sobre el menor. De esta forma, se puede observar cómo los valores mayores van subiendo hacia la parte superior de la tabla; de ahí el nombre de "método de burbuja" (Ver Código 2).

Para ir comparando un elemento con todos los demás necesitamos dos bucles anidados. En el bucle interior vamos comparando un elemento (bucle exterior) con todos los demás (bucle interior) e intercambiándolos si uno es mayor que el otro. Para ordenar de menor a mayor sólo es necesario cambiar el operando de condición "<" por ">":

Código 2. Método de la burbuja

```
Function ordenar_tabla()
 For n= 0 To 9
  For m= 0 To 9
   If Puntuacion(m) < Puntuacion(n)
    ; Intercambiamos valores
    Aux_Puntuacion=Puntuacion(m): Aux_Jugador$=Jugador$(m)
    Puntuacion(m)=Puntuacion(n)  : Jugador$(m)=Jugador$(n)
    Puntuacion(n)=Aux_Puntuacion: Jugador$(n)=Aux_Jugador$
   EndIf
  Next
 Next
End Function
```

Código 3. Insertamos un nuevo récord

```
Function Insertar_en_Tabla(j$,p)
 Aux_Puntuacion=p : Aux_Jugador$=j$
  For n=9 To 0 Step -1
   If Aux_Puntuacion>Puntuacion(n)
    Puntuacion(n)=Aux_Puntuacion: Jugador$(n)=Aux_Jugador$
    Return
   EndIf
  Next
End Function
```

3

Para insertar una nueva puntuación en la tabla se puede utilizar un método de inserción simple, donde el nuevo elemento desplaza al último de la tabla.

```
If Puntuacion(m) < Puntuacion(n)
➡ De Mayor a Menor
If Puntuacion(m) > Puntuacion(n)
➡ De Menor a Mayor
```

INSERTANDO UN NUEVO RÉCORD EN LA TABLA

Ya tenemos solucionado el tema de la ordenación de los récords de mayor a menor. Ahora solo nos queda utilizar un método para añadir un nuevo récord en la tabla. Para ello utilizaremos el método de inserción (Ver. "Inserción.bb") (Ver Fig. 3) (Ver Código 3).

Una vez insertado el nuevo récord, volvemos a ordenar la tabla por el método de burbuja.

Fundamentalmente, lo que hacemos es añadir un elemento (j$ y p) al final de la tabla desplazando el valor más pequeño de la misma. Así que tenemos que buscar la nueva posición del elemento empezando por el final. Si el número insertado ("Aux_Puntuación") es mayor que el de la tabla ("Puntuación(n)") colocamos el nuevo número en su lugar.

Si es encontrado, colocamos ahí el nuevo récord y salimos del bucle.



En el próximo número...

... aprenderemos un sistema para grabar y reproducir los movimientos de la cámara. Ideal para realizar presentaciones de nuestro juego con el motor del Blitz3D.

Juegos de rol o JDR (RPG)

En esta entrega vamos a centrarnos en otro de los grandes géneros del mundo de los videojuegos para ordenador: los juegos de Rol o JDR (RPG).

En este tipo de producciones, el jugador toma la identidad de un personaje o varios para participar en una aventura mágica. Hay infinidad de juegos de este género para PC, así que veremos solo los más representativos desde los comienzos.

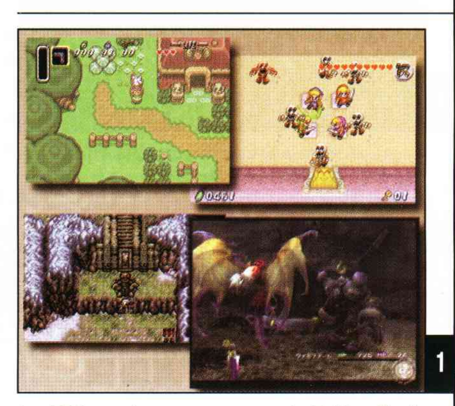
HISTORIA Y EVOLUCIÓN

Prácticamente todos los juegos de rol para ordenador siguen las reglas básicas *Dungeons & Dragons* (Dragones y Mazmorras) publicadas por Dave Amerson y Gary Gygax en 1974. Fueron creadas a partir de los *Wargames* clásicos que se jugaban con tablero, fichas y figuras de dragones, enanos y otras criaturas. La diferencia para estas nuevas reglas radicaba en que el jugador solo controlaba a un personaje y no a un ejército. En la década de los 80, los juegos de rol pasaron de los libros de lucha y tableros al ordenador para acercar al público en general a este género tan carismático. Los primeros juegos pasados a ordenador se basaban en texto sin ningún tipo de gráficos, algo parecido a como se haría con un libro. Sin embargo, aun avanzada la tecnología, el espíritu de los juegos de rol se ha mantenido intacto. Antes de que los ordenadores hubieran dominado el mercado lúdico, eran las consolas las que ofrecían todo tipo de géneros y es obligatorio hablar de los primeros RPG para consolas que siguieron las reglas *D & D* como *Quest For The Rings* publicado para Odyseey o *Cloudy Mountains* para Intellivision a comienzos de los 70. Al principio, estos juegos mezclaban la acción con ciertos toques de rol. Fueron Nintendo y Sega los que dominaron el mundo RPG con el lanzamiento de *Zelda* y la serie *Final Fantasy* con unos gráficos e historia increíbles para la época. Generalmente, la perspectiva usada era cenital pero había títulos que simulaban las 3D pantalla a pantalla como *Shining in the Darkness*. Las producciones para consolas siguen hoy en día con gran éxito, y quizás sea *Final Fantasy* la más famosa y aclamada. Sin embargo, el dominio del PC hizo que los RPG adquirieran otra dimensión en gráficos y jugabilidad. Aunque una de las producciones para ordenador que más caló en el público fue la serie *Ultima* creada por Richard Garriot en 1980. Garriot desarrolló 9 títulos, todos siguiendo los avances tecnológicos del momento. Desde la representación en blanco y negro de la primera entrega, pasando por la vista cenital clásica, hasta la representación 3D a partir de *Ultima Underworld*. Pero para clasificar a los RPG más sobresalientes de la historia para PC debemos guiarnos por los que siguen las reglas "Dungeons & Dragons" y los más avanzados "Advanced D&D".

DUNGEONS & DRAGONS Y ADVANCED DUNGEONS &DRANGONS

Las reglas *D &D* se basan en que el jugador encarna a un personaje y éste va aumentando de experiencia y poder durante el juego. Básicamente *AD & D* es igual que su antecesor pero incorporaba más reglas y subreglas triunfando a mediados de los 80.

Durante más de una década se


1

Los RPG mas famosos para consola: Zelda y Final Fantasy.


2

Diablo I y II. Clásicos indiscutibles del género.


3

(Arriba) Dungeon Siege, exquisito entorno 3D real. (Abajo) Nox intentó competir con Diablo sin conseguirlo.



Juegos de Rol modernos. (Arriba) Neverwinter Nights de Bioware con editor de juegos incluido. (Abajo) Morrowind: The Elder scrolls III, la última entrega de la serie The Elder scrolls.

LA BIOGRAFÍA...

RICHARD GARRIOTT

Creador de la serie **Ultima**

Después de vender 30.000 copias de su primer juego de rol *Akalabeth* para Apple II en 1980 crea el primer título *Ultima* y funda la compañía Origin System con su hermano Robert. Amante de los juegos de rol en todas sus vertientes, lucha por conseguir introducir a los jugadores en mundos de fantasía más allá de los prejuicios mundanos. Es uno de los desarrolladores más veteranos y respetados de la industria gracias a su trabajo con la serie *Ultima*. Llegó a convencer con las 3D de *Ultima IX Ascension*. Actualmente ya no está en Origin y está totalmente dedicado a *Ultima: Online 2* totalmente en 3D.



Richard Garriott.

ha intentado recrear la experiencia de los juegos *D&D* de lápiz y papel en ordenador. Fue la empresa *SSI* (Strategic Simulations Incorporated) la que primero logró realizar algo positivo y lo hizo *con Pool of Radiance* (1988) y más de una docena de títulos más como: *Heroes of the Lance* (1988), *Secret of the Silver Blades* (1990), el fantástico *Eye of the Beholder I* (1991), primer juego basado en *AD&D* en primera persona con un motor gráfico en pseudo tiempo real.

Siguieron otros títulos basados en la idea de *Eye of the Beholder* como: *Dungeon Hack* (1993) o *Menzoberrazan* (1994). Al margen de SSI, otras desarrolladoras destacaban por la calidad de sus juegos considerados clásicos del género como *SepterraCore: Legacy of the Creator* (Valkyrie Studios, 1999) o una de las producciones que consagró el género de la franquicia *AD&D* y que implantó una forma de hacer juegos de Rol fue *Baldur´s Gate* en 1998, publicado por InterPlay y desarrollado por BioWare. Anterior a estos títulos, una desarrolladora muy especial con un talento increíble y un equilibrado en los juegos inmejorable llamada Blizzard Entertaiment publica *Diablo*. Ambientado en el mundo medieval, sigue las reglas básicas de *D & D*. Era la primera incursión de Blizzard en Windows y poseía unos gráficos bien acabados con solo 256 colores. Su concepto de jugabilidad es simple y llega a multitud de usuarios de todo el mundo. Continuó la saga con *Diablo II* (2000). No tan original pero con una jugabilidad envidiable y con gráficos grandes y detallados. Los juegos de rol para PC cobran fuerza en el año 2000 con títulos de gran belleza gráfica. Clásicos como *Pool of Radiance* continúan su andadura por el universo de los Reinos Olvidados en 2001 con *Ruins of Myth Drannor*. Otra desarrolladora de renombre en los juegos de estrategia entra en el mundo del RPG con un título que pretendía desbancar el liderazgo de *Diablo*, nos referimos a Westwood Studios con *Nox* (2001). Fácil de jugar y con un motor gráfico más que aceptable casi alcanza su cometido. Ese mismo año Microsoft decide distribuir un título de *Gas Powered Games* que rompe moldes al combinar lo mejor de otros títulos de renombre y la potencia gráfica de las nuevas tarjetas de vídeo. Nos referimos a *Dungeon Siege*. Siguiendo las reglas *AD & D*, este título posee grandes dosis de acción, aventura y rol con gráficos fantásticos y completamente en 3D, y lo que es más importante; jugabilidad. De nuevo Bioware deja el mundo de las dos dimensiones con *Throne of Baal* y entra de lleno en las 3D con *Neverwinter Nights*, uno de los títulos más esperados basado en las reglas *D&D*. En este título poseemos total libertad de movimientos y podemos controlar varios personajes con habilidades diferentes. Además, viene incorporado de un editor de niveles y personajes fantástico. No podemos olvidar *Morrowind: The Elder scrolls III* (Bethesda Softworks/ Ubi Soft, 2002), la última joya del rol actual en primera y tercera persona.

RPG EN INTERNET

No queremos terminar sin hablar de las posibilidades que internet ofrece a los mundos de aventuras creados por los RPG. El género aprovecha las posibilidades de la red para perpetuarse dentro de mundos infinitos online. Es *Ultima Online* una de las que pervive actualmente con fuerza en todo el mundo a través de internet con miles de jugadores encarnando a personajes humanos en mundos de fantasía. Pero no fue *Ultima* el único título que llevó el rol a todo el mundo a través de la red. Encontramos también un título que ofrece al jugador una mayor libertad de movimientos, mejores gráficos y la posibilidad de encarnar a otros personajes no humanos: *Everquest* y *Everquest II*.



En el próximo número...

... hablaremos del género de aventuras gráficas.

Cuestionario Videojuegos 14

Preguntas

1. Modifica la luz ambiental de una escena y crea tres luces de diferentes tipos en Blitz3D.
2. ¿Cómo podemos modificar la intensidad y amplitud de un foco de luz en Blitz3D?
3. ¿Qué es un emisor de partículas?
4. Define una estructura básica para crear un sistema de partículas.
5. ¿Cómo obtenemos una plantilla o template de un modelo para pintar la textura con Paint Shop Pro?
6. ¿Cómo podemos obtener una textura desde Bryce?
7. ¿Cómo podemos insertar cinco compases en blanco dentro del pentagrama de una pista en Anvil Studio?
8. ¿Cómo podemos desplazar una nota del pentagrama hacia atrás o hacia delante en Anvil Studio?
9. Escribe la rutina para realizar una ordenación de veinte números por el método de burbuja simple.
10. Escribe una rutina para insertar un elemento en una tabla de veinte números.

Respuestas al cuestionario 13

▷ **1.** Utilizando la instrucción "CameraViewport":
CameraViewport PosX, PosY, Ancho, Alto

▷ **2.** Convertimos el objeto en "pickeable" con "EntityPickMode" y luego utilizando "CameraPick".

▷ **3.** Utilizando "EntityCollided", "CountCollisions" y "CollisionEntity":

```
If EntityCollided (bionave,ENTIDAD_ENEMIGO)
   For n=1 To CountCollisions(bionave)
      ent=CollisionEntity(bionave,n)
   Next
   For volador.tipo_voladores= Each tipo_voladores
      If volador\entidad_volador=ent  text 0,0,"DETECTADO": Return
   Next
EndIf
```

▷ **4.** Utilizando la instrucción "AlignToVector" y asignándole al sprite el modo 4 de visualización con "SpriteViewMode".

▷ **5.** Mediante el escalado y desplazamiento de los vértices de los polígonos.

▷ **6.** Para poder pintar el modelo en Deep Paint 3D, éste debe tener un mapeado UV y un material asignado.

▷ **7.** Debemos preparar las conexiones MIDI. Luego, asegurar que el programa recibe los eventos MIDI a través de las opciones de puerto MIDI de "Synthesizers".

▷ **8.** Seleccionando la opción "quantize Entire Track" del menú "Track".

▷ **9.**

```
Dim Letra$  (Num_letras)
Restore Alfabeto
For n=0 To Num_letras
   Read Letra$(n)
Next
.Alfabeto
Data "A","B","C","D","E","F","G","H","I","J","K","L","M"
Data "N","O","P","Q","R","S","T","U","V","W","X","Y","Z"
Data "1","2","3","4","5","6","7","8","9","0"
```

10. Utilizando una imagen para cada letra y sustituirla una vez hallada dentro de la frase:

```
For n = 1 To Len ( Frase$ )
   For m=0 To Num_letras
      If Mid(Frase$,n,1)=Letra$(m)
         GLetra=Grafico(m)
         DrawImage GLetra,x,y
      EndIf
   Next
Next
```

Contenido CD-ROM 14

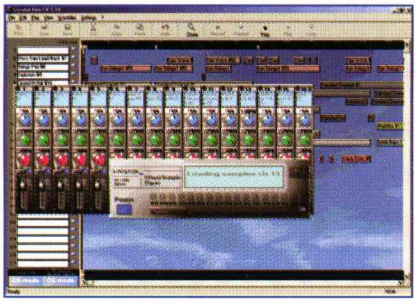
▶ AUDIO

■ Audio Phonics Guitar Tuner 1.02
Si aún compones usando una guitarra, gracias a este programa podrás afinarla con rapidez y exactitud.

■ Audio Phonics Instrument Tuner 1.02
Otro afinador de instrumentos, complementario al primero, rápido y preciso.

■ Da Metronome 1.1
Completo metrónomo por ordenador para ayudarte a llevar correctamente el ritmo en tus composiciones.

■ Locator MIDI Sequencer 1.10
Secuenciador MIDI de 256 canales. Tiene un teclado virtual, un mixer y un editor WAV con soporte de efectos Direct-X.

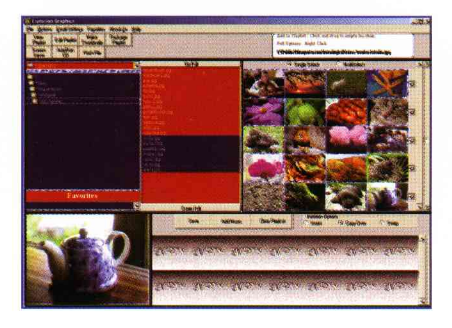


■ Cool MP3 Splitter
Divide MP3 por tiempo o tamaño y une los fragmentos.

■ Tactile12000
Aplicación basada en QuickTime que simula dos bandejas de DJ en 3D para mezcla de audio.

▶ DISEÑO 2D

■ Explosion Graphics 3.8.2
Una forma divertida de ver tus fotografías y de escuchar tus archivos de música.



■ Autoimager
Potente y completísimo procesador de imágenes en masa. Ideal para trabajar rápido con muchas imágenes.

■ Quick Screen Capture 1.2
Práctico y funcional capturador repleto de interesantes funciones.

■ EyeBatch 2.0
Renombra, retoca y procesa imágenes en grupo gracias a esta aplicación.

■ PhotoCleaner 1.3
Arregla los problemas comunes que puedas tener en tus imágenes con un solo clic.

■ PhotoPhilia 1.7
Organiza tus imágenes y aplícales sencillos efectos usando este programa.

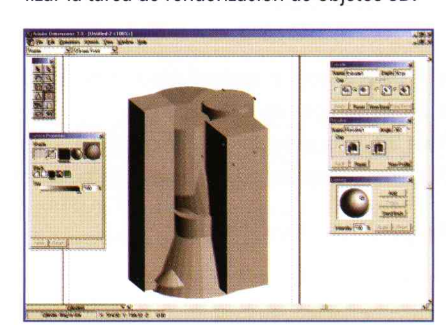
▶ DISEÑO 3D

■ 3D AIM Animation 3.0
Crea animaciones en tres dimensiones usando múltiples modelos.

■ 3D Art Composer 1.1
Construye imágenes en 3D a partir de distintos modelos y fondos.

■ Crossroads
Completo programa para convertir distintos archivos de 3D fácilmente.

■ Adobe Dimensions 3.0
Herramienta poderosa con la que podrás realizar la tarea de renderización de objetos 3D.



■ ModelPress 2.0.0.37
Publica en formato web archivos en tres dimensiones.

■ Terragen 0.8.11
Excelente programa para diseñar paisajes en 3D y renderizar escenas muy realistas.

▶ PROGRAMACIÓN

■ QSetup Installation Suite 3.0
Crea instalaciones sin necesidad de realizar scripts.

■ ImpulseStudio 3.05
Completa suite de componentes Active X para programar en Visual Basic.



■ VTune Performance Analyzer 6.1
Excelente utilidad para analizar datos de diversos modos.

▶ JUEGOS

■ Diablo
Uno de los juegos de rol más populares. Ambientado en la Edad Media, ha llegado a jugadores de todo el mundo.



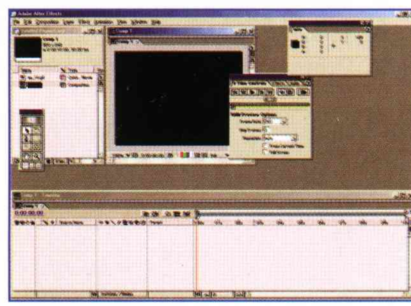
■ Final Fantasy
Uno de los RPG más famoso para consola, en esta versión para PC.

■ Dungeon Siege
Demo del popular juego con exquisito entorno 3D real. Te entusiasmará.

■ Zone of Fighters
Como todas las semanas, nuestro juego.

▶ VÍDEO

■ Adobe AfterEffects 5.5
La herramienta esencial para realizar fantásticos efectos visuales.



■ NVDVD 2.0
Excelente programa de vídeo para que tengas una auténtica experiencia multimedia.

▶ EXTRAS

En este apartado encontrarás todos los ejemplos de los que hablamos en el coleccionable, para que no pierdas detalle.